Spatialized Audio Streaming for Networked Virtual Environments

Roger Zimmermann
School of Computing
National University of Singapore
Singapore 117590
rogerz@comp.nus.edu.sg

Ke Liang
School of Computing
National University of Singapore
Singapore 117590
liangke@comp.nus.edu.sg

ABSTRACT
Networked virtual environments (NVE) are increasingly popular and represent a range of applications. Some online virtual worlds have a dedicated purpose, such as Massively Multiplayer Online Games (MMOG), while others implement more of foundational frameworks which are not necessarily applications per se, but form platforms to create applications. One of the premier examples of the latter is Second Life from Linden Lab. While such networked virtual environments show great potential for interesting and novel applications, many technical challenges remain. A significant shortcoming in current systems concerns the communication between virtual world participants. Early designs only supported text chat, while more recently voice chat has been introduced. However, often the voice communication paradigm still mimics traditional audio group conferencing without adaptation to the three-dimensional metaverse paradigm. In this study we present our design of an interactive audio streaming protocol that aims to enable the creation of an aural soundscape around the user that matches the visual experience. To make our design practical, scalable and to avoid further burdening the virtual world servers, our protocol is based on a peer-to-peer distribution topology. Simulation results are presented that show the feasibility and utility of our design.

Categories and Subject Descriptors
H.4.3 [Information Systems Applications]: Communications Applications—Computer conferencing, teleconferencing, and videoconferencing; H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Audio input/output

General Terms
Algorithms, Measurement, Performance

Keywords
Proximity audio, spatial audio, peer-to-peer streaming

1. INTRODUCTION
The Internet has become an indispensable tool for people to interact on a global scale. One type of large-scale interactive applications that is emerging on the Internet are networked virtual environments (NVE) where users can move a representation of themselves (also known as an avatar) in a shared virtual world and interact with each other. One of the most well-known examples of an NVE is Second Life (www.secondlife.com). Such virtual worlds are interesting for a number of reasons. Most importantly, some of these virtual environments are not applications per se, but they form the foundation for the creation of specific applications. For example, Second Life has been used for virtual meetings, training and recruitment by large corporations such as IBM. As a consequence, the term metaverse has also been used to describe such generic NVEs. Of course, one of the biggest application areas for NVEs are games, also termed Massively Multiplayer Online Games (MMOG).

NVEs are an active field of research and present a plethora of technical challenges. Among them are scalability of network traffic and server environments, end-to-end delay of user interactions, human-computer interface issues, visual representations and more. For some of these problems considerable progress has been achieved. For example the visual quality of the three-dimensional environments in these shared worlds is pleasing and constantly improving through better hardware and software algorithms. However, one area that has been sorely lagging behind is natural audio communication. The most widely used mode of communication in these shared worlds is through text chat. While this is a mature technology that is relatively easy to implement, it lacks much of the natural and immersive characteristic that good-quality voice communication could provide for its users. A number of voice communication approaches have been proposed in the literature and some commercial systems exist (e.g., offerings from Ventrilo and TeamSpeak). Existing solutions, with few exceptions, fall short in that they do not provide a three-dimensional aural experience that matches the visuals.

In this study we present the design of a voice communication platform for NVEs that is massively scalable and reduces the need for large, centralized servers. It aims to be an add-on component for popular virtual worlds and games (for example Second Life and World of Warcraft). The proposed approach provides voice services to very large user groups. It achieves this by (1) implementing a peer-to-peer architecture that distributes network traffic and audio processing, and (2) by providing proximity audio, an audio distribu-
tion mode that allows for a natural and automatic control of group sizes in online virtual environments.

To achieve natural, immersive interaction for virtual world participants two important concepts need to be realized. First, communication should naturally be enabled within a certain proximity of an avatar, i.e., users are more likely to interact within a close distance range measured in virtual coordinates. This concept, which is commonly referred to as Area of Interest (AoI), has been well studied [16, 19]. Once the AoI is defined, the users within the same AoI are expected to be clustered to reduce delay and control message overhead.

The second component for natural interaction is the creation of an aural soundscapes that matches the visual landscape of an environment. Spatialized, three-dimensional audio rendering is now possible on commodity personal computer hardware. While surround-sound speakers (e.g., 5.1 channels) can provide full 360 degree rendering, even stereo speakers are capable of creating roughly a 180 degree sound field in front of the user. Libraries such as OpenAL implement the signal processing algorithms necessary to position sound sources in specific locations. In a client-server architecture all audio sources are sent to a centralized processor which creates, for example, a customized 3D stereo mix to be sent to each user. In a peer-to-peer architecture the approach is different altogether. In our design the audio streams from different sources are kept (mostly) separate until rendered at the client. The advantage is that no central mixer with large resource requirements (bandwidth and processing capabilities) is necessary and the spatializing algorithms run directly on the client. Because the bandwidth capacity of each node in a peer-to-peer network to receive and deliver data is limited and heterogeneous, a challenging problem in this design is how to fully utilize the limited and heterogeneous network resources to deliver many separate audio streams and at the same time, to limit the overlay stream delay. However, as we shall see, the AoI management naturally constrains the distribution area of the streams.

To summarize, our design presented in this study features the following contributions:

- The implementation of proximity and spatialized audio allows for the automatic and natural control of audio volume and directionality in virtual environments.
- The system implements realistic in-coming and outgoing bandwidth limits for each client (i.e., the in/out degree-limits within the distribution trees). We consider practical, asymmetric broadband connections where only a fraction of the bandwidth should be used for voice communication.
- For avatars situated in crowded groups, multiple audio streams may be mixed together to reduce the incoming streaming rate and adhere to pre-set bandwidth limits. In essence, shared branches of distribution trees may be merged under specific conditions.
- The system is largely de-coupled from the game control architecture (which distributes, for example, the game state information). Therefore it can be integrated into existing or future networked virtual worlds with little effort, irrespective of whether the design for the state control traffic is centralized or distributed. In our design the only critical information required from the game servers is the virtual coordinates of the avatars.

The remainder of this manuscript describes our approach in detail. We will start with a survey of the related work and techniques in Section 2. A detailed introduction of the design of our approach is contained in Section 3. Section 4 describes the evaluation of our techniques and presents simulation and performance results in comparison with some existing algorithms. Finally, Section 5 draws conclusions and outlines future work in this space.

## 2. RELATED WORK

Recent interest in metaverses and virtual worlds has resulted in significant research being conducted in the area of application-level game infrastructures for Massively Multiplayer Online Games, MMORPG [1, 5, 6, 9, 10, 12, 18, 19]. The currently favored design for transmitting game state information in NVEs is to use a client-server architecture that utilizes server clusters which partition the game world into many different game zone regions (often based on a grid layout). Each game zone is handled by one server machine to achieve load balancing across the cluster in terms of computations and network game traffic. Regions may be statically or dynamically assigned to servers [1].

One natural inclination is to extend the same client-server architecture that is used for the interactive game traffic to support voice communication. An alternative approach is to

### Table 1: Categorization of interactive audio streaming techniques and some exemplar systems.

<table>
<thead>
<tr>
<th>Approach proposed in this study</th>
<th>Monophonic Audio</th>
<th>Spatialized (3D) Audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client-Server or Proxy</td>
<td>Ventrilo, TeamSpeak</td>
<td>DiamondWare (used by Vivox), DICE [3]</td>
</tr>
<tr>
<td>Peer-to-Peer</td>
<td>Skype, AudioPeer [20]</td>
<td>Approach proposed in this study</td>
</tr>
</tbody>
</table>
use a peer-to-peer distribution architecture. We can further differentiate between voice communication systems that provide monophonic audio and those that support spatialized, proximity audio. Table 1 categorizes existing voice streaming systems into four quadrants along two axes: client-server (or proxy) versus peer-to-peer, and monophonic versus spatialized audio. We will briefly describe some representative work of each quadrant.

**Client-Server & Mono Audio.** From a technical perspective, this category represents the most traditional approach. For NVE applications, since the user is already situated in front of a computer, voice-over-IP (VoIP) systems are the natural choice. Some such systems are completely separate from any games, while others are targeted towards virtual worlds and provide some form of integration (e.g., the commercial offerings from Ventilio and TeamSpeak).

**Peer-to-Peer & Mono Audio.** Peer-to-peer application-level streaming topologies have become popular and successful academic prototypes and commercial systems have been built. One natural benefit of p2p is that each participating node contributes resources such as memory, bandwidth and processor cycles, which allows for very scalable designs. Some existing systems in this category are completely separate from any NVE application (e.g., Skype, AudioPeer [20, 11]), while others are targeted towards virtual worlds and provide some form of integration. In a p2p design, distributed stream mixing needs to be performed [8]. Suitable techniques to handle distributed audio mixing have been proposed in the literature [15]. Several P2P overlay multicast designs have been evaluated by Castro et al. [4].

From the perspective of NVEs, the most important shortcoming of these monophonic systems is that they do not take advantage of the location, distance or the directionality between user avatars in the virtual world. Therefore, the aural environment does not match the visual perception of the user and—while functional—does not provide a seamless experience. Consequently, the work that is more related to our approach is concerned with providing an immersive, 3D aural environment. Spatialized audio rendering is now possible on commodity personal computer sound cards and is widely used for local (i.e., non-streaming) game sound designs. Libraries such as OpenAL allow for efficient rendering implementations and interest is growing to use spatialized sound in interactive communication applications.

**Client-Server & 3D Audio.** With a client-server topology, 3D audio rendering is supported by capturing mono streams at each client and forwarding them to the central server for processing. There, complete information about all avatar locations is available and a personalized 3D sound mix is rendered and packed into a stereo stream to be sent back to each client. However, with this approach significant audio bandwidth and computational power for signal processing is required at the server. This can easily place an undue burden on the already taxed game server infrastructure. Boustead and Safaei [2] have surveilled different delivery architectures. They concluded that some form of distribution would be beneficial. To remove the central server bottleneck some proxy-based solutions [14] have been proposed to load-balance the bandwidth and processing load (e.g., DICE [3]). A commercial product in this space is offered by Diamond-Ware (which has been licensed by Vivox).

**Peer-to-Peer & 3D Audio.** To the best of our knowledge there exists no comparable platform to the fully decentralized, peer-to-peer architecture that we are proposing for spatial audio streaming. The closest variants are the proxy-based solutions mentioned in the previous paragraph. We expect that improved audio capabilities will become increasingly important to catch up with the visual developments of virtual worlds as more academic, commercial and other entities seek to build multimodal shared virtual spaces for person-to-person interaction in mass populated online communities.

### 3. DESIGN

#### 3.1 Objectives and Technical Challenges

Our goal is to design a scalable streaming architecture that is capable of providing 3D spatial audio services in NVE-type applications. Specifically we aim to satisfy the following technical challenges.

**Spatialized (Immersive) Audio.** Audio streams from different sources are rendered according to the spatial and directional relationship between receiver and senders in the virtual world to create an immersive audio experience. Essentially, an avatar’s physical movements naturally control the audio stream distribution through the underlying streaming topology. An avatar can stroll through the virtual setting listening to conversations of interest and speaking to other avatars when desired.

**Low Latency.** The end-to-end delay from the source (sender) to the receiver(s) needs to be minimized so that meaningful interaction can occur. The delay performance of an overlay multicast tree is determined by several factors such as link quality, topology configuration, bandwidth capacity. A streaming architecture must carefully optimize the delay based on all these different parameters.

**Scalability.** Spatialized voice traffic is voluminous and may consume more bandwidth compared with other types of game traffic. With traditional voice conferencing only a single, mono channel is required and the voices of different participants can be mixed together at appropriate node locations within the network. However, proximity audio requires that multiple voice channels be kept separate until they are rendered and merged into a three-dimensional aural voice field on the participant’s client computer. Our approach is to use a peer-to-peer based design that distributes the streaming load to the participating nodes and hence bypasses the game servers.

The design of an architecture for spatial audio streaming presents formidable challenges. However, we feel that a successful solution will enable capabilities beyond what is in existence today. The shared space will much more realistically imitate a physical setting by adjusting sound, volume and directionality according to avatar proximity and position. Because the audio distribution within the virtual environment is based on proximity and hence localized, a massive scale is possible that cannot be achieved with traditional voice conferencing. It is our hope that we will not only gain important insight into many of the technical challenges, but also that our system design will be the basis to study future communications concepts and interaction paradigms.

Before we detail our streaming architecture we will briefly review the relevant concepts of area of interest (AoI) modeling for the hearing range and spatialized audio rendering as they relate to our approach.
3.1 Area of Interest Modeling

In NVE-type applications, users command their avatars to move around the shared virtual world. The area of interest (AoI) of a user is usually defined as the space around the user in this shared virtual world where interaction, either between users or between user and the world, is likely to happen. Previous research has explored the temporal and spatial locality for interaction among users in large-scale systems [16, 19]. It has been shown that users are more likely to interact with each other when their positions are close in the virtual world. We believe that this observation extends to streaming services in NVE-type applications, i.e., users who share their AoIs are also more likely to stream from each other. In our audio streaming application scenario, the AoI corresponds to the hearing range of each avatar. We currently model the hearing range as a circular area centered at the user’s position. In reality large objects (such as walls, buildings, etc.) may obstruct the audio field. However, we will use the simplified, circular model initially and expand our work to include a more accurate sound environment in the future. We currently do not explicitly model different levels of audio volume within the AoI. The audio volume could be attenuated towards the edge of the hearing range. This can be achieved in the rendering engine, based on the distance information that is available for each audio source.

Another aspect of practical importance is that most large virtual worlds are partitioned into cells of some shape (e.g., rectangles or hexagons), each handled by a server to provide load balancing and scalability. The cell edges act as natural boundaries where the visual AoI is truncated. The spatial audio streaming solution will use the same avatar position information and cell boundaries as the visuals, and therefore the audio-AoI will also be pruned accordingly.

3.1.2 Spatialized Audio Rendering

Audio rendering to position sound sources across a spatial volume is now available on commodity personal computers. Such capabilities are usually contained in software libraries that can take advantage of different sound card hardware and transducer setups. For example, audio may be rendered via 2, 5.1, 6.1, 7.1 speakers, or via headphones. Most existing 3D sound processing modules require that the audio from different sources is kept separate until rendered. There are algorithms to re-generate 3D immersive audio from only one or two audio input channels, but the results are not very satisfactory. Because the send and receive network capacity for each node in a peer-to-peer network is limited and heterogeneous, it becomes very challenging to utilize the network resources effectively to deliver several separate audio streams while at the same time keep the overlay delay bounded.

3.2 System Model

The environment that we are considering operates on two different planes (see Figure 1). The physical space consists of routers connected by links. The end systems are connected to this network at different points through access links. The overlay network is modeled as a graph $G(V, E)$, where $V$ is the set of vertices and $E = V \times V$ is the set of edges. Each vertex $v \in V$ represents a user or end system. Every edge $e \in E$ denotes the unicast path between the two end vertices of the edge.

The virtual space is represented as a 2D square of side-length $l$ and formalized as a dynamic graph. Each avatar uses coordinates $(x, y) \in [0 \ldots l, 0 \ldots l]$ to represent its location in the virtual world. The distance between two avatars is defined as the Euclidean distance between them. There is a one-to-one mapping between users in the physical space and avatars in the virtual space. We use the term node to refer to both a user in the physical space and an avatar in the virtual space. In the rest of the manuscript we will use the terms node, user and avatar interchangeably. Node $i$ is denoted with $n_i$. For identification purposes each node has a unique global id in the virtual space. The AoI (i.e., hearing range) of an avatar is represented as a circle centered at the avatar’s position, with the AoI radius being equal for all avatars. An avatar $n_i$ within an avatar $n_0$’s AoI is termed a neighbor of $n_1$ (see Figure 2 for examples). Motion and position information of avatars are updated and distributed via the regular game control infrastructure.

The access link of each node can sustain a certain upload and download bandwidth (which usually is asymmetric). In our system we assume that audio communication may only use a certain fraction of the available bandwidth and that the rest is allocated to the game traffic. Each audio stream $AS_i$ consumes bandwidth at a rate of $b$ (we use $b = 25$ kbps in our experiments) and we compute the indegree $db_{in}^i$ and outdegree $db_{out}^i$ of each node $i$ as the number of audio streams it can send to or receive from other nodes.

Our goal is to form a distribution topology through which each sender can successfully reach all of the receivers within its AoI. Multiple topologies have been suggested for this purpose in the past, for example meshes, rings, trees, and multi-trees. One important objective of our design is to respect the in- and outdegree limits of each node. Oftentimes, existing work on peer-to-peer streaming either considers artificial or no bandwidth limits for end systems. Tree-structures provide a good compromise between the maximum path length (which affects the end-to-end latency) and the out-going bandwidth required at each node. We will therefore focus our design on spanning trees. The main challenge is that many variants of degree-constrained minimum spanning trees (MST) have a high computational complexity (or are $NP$-complete). Next we will state the problem more formally and then investigate solutions that are augmented with heuristics.
### 3.3 Problem Formulation

We aim to solve a many-to-many audio streaming problem where every node in the overlay network could be the source of one audio stream and the receiver of one or several others. Since the set of receivers is (usually) different for each sender, an applicable solution is the construction of a forest of multicast trees, with the trees rooted at all the senders. Furthermore, the indegree and outdegree of every node in the overlay network is bounded.

Therefore, the problem is defined as constructing a forest of degree-bounded minimum latency multicast trees:

Given an undirected graph $G = (V, E)$ representing the virtual space, degree bounds $d_{in}^n$ and $d_{out}^n$ for each node $i$ in $G$, and a set of senders $S$ in $G$. Construct a minimum latency multicast tree $T_i$ for each sender and the set of neighbors within its AoI such that the maximum number of successful receivers $R_i^s$ is achieved.

Maximize successful receivers : $|R^s(S)| = \sum_{i \in S} |R_i^s|

Minimize latency : $\frac{1}{|R^s(S)|} \sum_{j \in S} \sum_{k \in R_j} L(j, k)$

Subject to degree constraints : $d_{in}^s \leq d_{in}^v, v \in V \quad (1)$

$\quad d_{out}^s \leq d_{out}^v, v \in V \quad (2)$

Successful receivers are those neighbors of a sender that can receive the stream. $R_i^s, i \in S$ is the set of successful receivers in the multicast tree rooted at sender $i$. $R^s(S)$ denotes the set of all successful receivers in the virtual space $G$. It should be noted that one node might be a successful receiver of multiple multicast trees (e.g., node $n_3$ in Figure 2). The in- and outdegrees used by node $v$ are denoted $d_{in}^v$ and $d_{out}^v$, respectively. $L(j, k)$ refers to the latency of the path from node $j$ to node $k$.

### 3.4 Approach

Finding an optimal solution to the multi-objective problem stated above is very challenging. Combining the different objectives into one scalar goal and finding a Pareto optimal point may not be feasible due to the computational complexity that is introduced by the degree bounds. Therefore, our proposed method is to find a solution via a multi-stage process.

Fortunately, the streaming of audio samples has a characteristic that can help us to reduce conflicts in the in- and outdegree usage. Recall that due to degree constraints not all candidate receivers may be able to acquire a stream. Multiple streams may be merged (i.e., mixed) together by combining their digital sample values. The resulting stream requires the same bandwidth as each of the input streams. As shown in Figure 3, node $n_a$ can mix two streams ($AS_v$ and $AS_m$) together and send the mixed stream ($AS_v + AS_m$) to node $n_b$. In this way, one outdegree of node $n_a$ and one indegree of node $n_b$ are saved while node $n_b$ can still render the two streams.

![Figure 3: Audio streams may be mixed to save bandwidth.](image)

Two important issues arise with stream mixing. First, the mixing process is irreversible which means the combined streams cannot be split again. As a result, streams should only be mixed under specific circumstances which will be described in a subsequent section. Second, some spatial information loss may occur due to the mixing process. Recall that each audio source is associated with a sender location in the virtual space. When merging two sound streams their spatial locations also must be combined. To minimize the error one may associate the merged stream with a ‘phantom’ source located in the middle between the two original sources. We will quantify the spatial information loss of our stream merging approach in the experimental section. Note that spatial information loss of a closely positioned source may be more noticeable than for a source located at a greater distance. However, more investigation is necessary to quantify such detailed effects adequately.

With the addition of the stream mixing procedure we now have three optimization goals for our system: (1) maximizing the number of successful receivers, (2) minimizing the latencies of these receivers and (3) minimizing the spatial information loss when some streams are mixed. Our method to achieve these goals and create a forest of multicast trees is divided into two stages. The first process executes on sender $i$ with three sub-phases. Subsequently the second process is started on the set of receivers within the AoI of sender $i$.

**Stage 1: Local Multicast Tree Construction and Optimization** (executed at the sender).

Stage 1 has three phases with the goal of creating a local multicast tree $T_i$ rooted at sender $i$ while respecting the de-
gree limits of all the nodes within the AoI of \( i \). During this process the nodes are partitioned into three disjoint sets: Core\(_i\), Leaf\(_i\), and \( U_i \). The set Leaf\(_i\) contains all the leave nodes of \( T_i \), and Core\(_i\) contains \( \{ T_i - \text{Leaf}_i \} \). The set \( U_i \) contains all the receivers that were unsuccessful in connecting to \( T_i \) due to exhausted degree constraints (see Figure 4 for an example). At the end of this stage the nodes in sets Leaf\(_i\) and \( U_i \) will be notified to execute the Stage 2 process.

**Stage (2): Audio Stream Mixing** (executed at the receivers).

During this stage, unsuccessful receivers and leaf nodes in the local multicast tree \( T_i \) can select nodes in the core of \( T_i \) to mix streams for them. The goal is to increase the number of successful receivers. If there are two or more nodes in the core that can mix streams for the unsuccessful node, the one which results in the least spatial information loss is chosen.

Next we will elaborate on the details of each stage.

### 3.4.1 Local Multicast Tree Construction and Optimization

We assume that state information is exchanged periodically among nodes to gather information about neighbors (i.e., a latency matrix, available outdegrees and indegrees). Being aware of state information of its neighbors, each node can construct a local multicast tree using three sub-phases of Stage 1.

Illustrated in Algorithm 1, the local multicast tree construction process works as follows. In the first phase, all nodes that have no more available indegrees since they are participating in other multicast trees are put into the unsuccessful receiver set \( U_i \), as shown in lines 1-6. During the second phase, indicated by line 8, a local multicast tree is constructed based on the remaining nodes, and rooted at the sender. Here we employ a traditional, fast algorithm such as the Minimum Spanning Tree (MST) or the Minimum Latency Tree (MLT) method. Note that these standard algorithms do not respect the in- and outdegree limits. Therefore, we slightly modify the algorithms such that they stop the tree construction whenever a degree constraint would be violated by proceeding on that branch. After this step, not all candidate receivers within the AoI are able to connect to the tree. Therefore, we propose a heuristic third phase called MulticastTreeAdjust (line 9) to maximize the number of successful receivers. More details on this last phase are given in Algorithm 2 and in the paragraphs below. The output of the Stage 1 process is a partitioning of all the nodes within the AoI of sender \( i \) into three sets: Core\(_i\), Leaf\(_i\), and \( U_i \).

**Figure 4: The two stages of our proposed approach.**

---

**Algorithm 1** Stage 1: Multicast Tree Construction and Optimization

**Require:** \( v, N_v, A_v^{\text{in}}, A_v^{\text{out}}, L_v \)

**Ensure:** \( U_v, \text{Leaf}_v, \text{Core}_v, M_i \)

1: \( U_v \leftarrow \emptyset \)
2: \( M_v \leftarrow \emptyset \)
3: for each \( i \in N_v \) do
   4:     if \( A_v^{\text{out}}[i] = 0 \) then
   5:         \( U_v \leftarrow \{ i \} + U_v \)
   6: end if
7: end for
8: \( T_v \leftarrow \text{MulticastTreeConstruct}(v, L_v, N_v - U_v) \)
9: MulticastTreeAdjust\( (T_v, A_v^{\text{in}}, A_v^{\text{out}}) \)

---

In order to achieve low latency we construct a minimum latency tree (MLT), constructed by combining all shortest paths from the sender to its neighbors in line 8 of Algorithm 1. It ensures all the successful receivers observe the minimum latency and fast algorithms for MLT exist [7]. The time complexity of MLT is \( O(N \log N) \), where \( N \) is the number of nodes in the local multicast tree. In the experimental evaluation section we also use an MST for comparison purposes.

Our heuristic method used in the MulticastTreeAdjust (line 9) algorithm is based on computing a contribution value (CV) for each node, reflecting the maximum number of vertices that can be reached in its subtree while observing the outdegree bounds. As illustrated in Algorithm 2, we use a depth-first search to assign a CV to each node in the local multicast tree. Each node forwards its streams to child nodes which have a high CV. The run time complexity of MulticastTreeAdjust is \( O(N \log N + 3N) \). The loops of lines 1-4 and lines 17-23 require \( O(N) \) time. ComputeCV (shown in Algorithm 3) takes time \( O(N \log N + N) \) steps. Lines 6-16 also take time \( O(N) \). The total time complexity of the local multicast tree construction is \( O(2N \log N + 3N) \).

**Algorithm 2** MulticastTreeAdjust\( (T_v, A_v^{\text{out}}) \)

1: for each \( i \in V[T_v] \) do
2: \( CV[i] \leftarrow 0 \)
3: visit\( [i] \leftarrow 0 \)
4: end for
5: ComputeCV\( (v, T_v, CV, A_v^{\text{out}}) \)
6: for \( i \leftarrow v \) to \( T_v[T_v.size() - 1] \) do
7: if \( A_v[i] < T_v[i] \) then
8: \( \text{count} \leftarrow T_v[i].size() - A_v[i] \)
9: while \( \text{count} > 0 \) do
10: \( u \leftarrow T_v[i][T_v[i].size() - 1] \)
11: \( T_v \leftarrow T_v - \{ u \} - \text{Desc}_v(u) \)
12: \( U_v \leftarrow U_v + \{ u \} + \text{Desc}_v(u) \)
13: \( \text{count} \leftarrow \text{count} - 1 \)
14: end while
15: end if
16: end for
17: for \( k \leftarrow v \) to \( T_v[T_v.size() - 1] \) do
18: if \( T_v[k].size() > 0 \) then
19: \( \text{Core}_v \leftarrow \text{Core}_v + \{ k \} \)
20: else
21: \( \text{Leaf}_v \leftarrow \text{Leaf}_v + \{ k \} \)
22: end if
23: end for
3.4.2 Audio Stream Mixing

When a node becomes a speaker, it will send the audio stream through the local multicast tree it maintains. Due to the in/outdegree bounds of its neighbors, some may not be able to receive all their streams. To initiate Stage 2, the sender will broadcast a message to trigger the mixing node selection process among the currently unsuccessful receivers and leaf nodes in the multicast tree. The objective is to further increase the number of successful receivers while at the same time saving network resources. A drawback of this approach is that stream mixing will cause some spatial information loss for some nodes. However, we conjecture that it is generally preferable to receive a stream, albeit slightly misaligned in space, compared to no stream at all.

Rule 3.1. Let nb be a node in Leafy or Uy of sender nx. Let nx be a node in Corey. The stream sent by nx can be mixed at nx if \( \exists \text{m}, m_{\text{m}} \in T_{\text{m}} \wedge m_{\text{m}} \neq n_{\text{x}}, \) such that \( n_{\text{x}} \in \text{Core}_{y} \wedge n_{\text{b}} \in T_{\text{m}}[n_{\text{m}}] \wedge n_{\text{b}} \in \text{Leaf}_{m}. \)

Figure 5: Four conditions for simple stream mixing.

Rule Explanation. For \( n_{\text{x}} \) to be able to mix the stream \( A_{m} \) (sent by \( n_{\text{b}} \)) for \( n_{\text{x}} \), the following conditions should be satisfied (as shown in Figure 5): (1) \( n_{\text{x}}, n_{\text{b}} \in T_{v} \wedge n_{\text{x}}, n_{\text{b}} \in T_{m}, n_{\text{m}} \neq n_{\text{x}}. \) That is, both \( n_{\text{x}} \) and \( n_{\text{b}} \) must be members of two local multicast trees \( T_{v} \) and \( T_{m} \), since a mixing operation requires at least two streams. (2) \( n_{\text{x}} \in \text{Core}_{v} \wedge n_{\text{a}} \in \text{Core}_{m}, \) That is, \( n_{\text{x}} \) must be a non-leaf node in both trees since it will be forwarding the mixed stream. (3) \( n_{\text{a}} \in \text{Leaf}_{v} \wedge n_{\text{b}} \in \text{Leaf}_{m}. \) On the other hand, \( n_{\text{a}} \) must be a leaf node in both trees. This is desirable because if \( n_{\text{b}} \) is not a leaf node, its descendants may not be in the AoI of \( n_{\text{x}} \) and/or \( n_{\text{m}} \), which means they should not receive the mixed streams. (4) \( n_{\text{a}} \) must be a direct descendant of \( n_{\text{x}}. \) Again, this is desirable because any intermediate nodes are not guaranteed to be within the AoI of \( n_{\text{v}} \) or \( n_{\text{m}}. \)

These four conditions allow for a quick determination of which nodes are candidates for audio mixing. Additional cases exist, for example there could be intermediate nodes between \( n_{\text{x}} \) and \( n_{\text{b}} \), as long as they are within the AoI of both trees. However, the condition-checks necessary for the more extended cases require more processing time.

In our current system we use the four described conditions such that every \( n_{\text{x}} \) in \( U_{v} \) (with \( n_{\text{x}} \) as the sender) will search for a node in \( \text{Core}_{v} \) which can mix streams for it. Furthermore, if two or more nodes in \( \text{Core}_{v} \) can accomplish the mixing, the node which will result in less spatial information loss is chosen. As shown on the right of Figure 4, node \( n_{\text{x}} \) can mix streams for node \( n_{\text{a}} \) and node \( n_{\text{b}} \) can mix streams for node \( n_{\text{b}} \). As a result, the number of successful receivers is increased while network bandwidth is saved.

4. EXPERIMENTAL EVALUATION

To evaluate the feasibility and performance of our proposed approach we have implemented a comprehensive simulation environment based on the Boost Graph C++ Library [17]. Understanding the perceptual implications of the algorithm dynamics are of course also an important issue. We plan to investigate these aspects through a comprehensive user study once we have a fully operational prototype implementation. In our initial evaluation we report on the results of a simulated virtual world space with a 1,200 x 1,200 units square area, populated with 1,000 avatars (simulation parameters are listed in Table 3). As illustrated in Figure 6, we used three different node (avatar) location distributions in the virtual world space to explore various crowd-densities and characteristics: uniform, lightly clustered and significantly clustered.

To account for the delays in the network we randomly selected a latency value for each link from the distribution shown in Figure 7. The round-trip time (RTT) along a certain Internet path can be modelled with a shifted Gamma distribution [13]. That is, the latency between any two nodes can be defined as a random variable X which is gamma-distributed with scale \( \theta \) and shape \( k: X \sim \Gamma(k, \theta). \) We use

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual space</td>
<td>1,200 x 1,200 sq units</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>1,000</td>
</tr>
<tr>
<td>Aol of nodes</td>
<td>20 – 100 units</td>
</tr>
<tr>
<td>Talking probability of each node</td>
<td>0.1 – 0.8</td>
</tr>
<tr>
<td>Audio bandwidth</td>
<td>25 kbps</td>
</tr>
<tr>
<td>Link latency</td>
<td>see Fig. 7</td>
</tr>
<tr>
<td>Node bandwidth</td>
<td>see Fig. 8*</td>
</tr>
<tr>
<td>Distribution tree algorithms</td>
<td>MLT, MST, HP</td>
</tr>
</tbody>
</table>

*Used 50% of up- and 20% of downlink capacity.

Table 3: Simulation parameters.

Figure 6: Two extended cases require more processing time.

Figure 7: The round-trip time (RTT) along a certain Internet path can be modelled with a shifted Gamma distribution [13]. That is, the latency between any two nodes can be defined as a random variable X which is gamma-distributed with scale \( \theta \) and shape \( k: X \sim \Gamma(k, \theta). \) We use

Figure 8: Three different node (avatar) location distributions in the virtual world space to explore various crowd-densities and characteristics: uniform, lightly clustered and significantly clustered.

Figure 9: The round-trip time (RTT) along a certain Internet path can be modelled with a shifted Gamma distribution [13]. That is, the latency between any two nodes can be defined as a random variable X which is gamma-distributed with scale \( \theta \) and shape \( k: X \sim \Gamma(k, \theta). \) We use

The example shown in Figure 4 illustrates the overall process. All the nodes within the AoI of sender \( n_{x} \) will be partitioned into three sets: \( \text{Core}_{v}, \text{Leaf}_{v} \) and \( U_{v}. \) When line 7 in Algorithm 1 is executed, \( \text{Core}_{v} = \{n_{a}, n_{y}, n_{z}\}. \) \( \text{Leaf}_{v} = \{n_{a}, n_{y}, n_{f}, n_{y}\}. \) \( U_{v} = \{n_{a}, n_{l}\}. \) Each node in \( \text{Core}_{v} \) and \( \text{Leaf}_{v} \) will be assigned a CV by Algorithm 3. Node \( n_{x} \) only has 1 available outdegree but which has two receivers, nodes \( n_{p} \) and \( n_{c}. \) It will choose the node which has the largest CV, node \( n_{y} \) in the example, as its receiver, and node \( n_{p} \) will be moved to \( U_{v}. \)

Algorithm 3 ComputeCV(visist, s, T, CV, A_{v}^{out})

1: \( \text{visist}[s] \leftarrow 1 \)
2: if \( T_{v}[s], \text{size}() = 0 \) then
3: \( \text{return} \)
4: end if
5: for each \( i \in T_{v}[s], \) and \( \text{visist}[i] = 0 \) do
6: \( \text{ComputeCV}(\text{visit}, i, T_{v}, A_{v}^{out}, \text{CV}) \)
7: \( \text{ChildCV} \leftarrow \{\text{CV}[i]\} \)
8: end for
9: Sort \( T_{v}[s] \) in descending order of \( \text{CV} \)
10: \( \text{count} \leftarrow \text{Min}(T_{v}[s], \text{size}(), A_{v}^{out}[s]) \)
11: if \( \text{count} \neq 0 \) then
12: \( \text{for each } j \leftarrow \text{to count do} \)
13: \( \text{CV}[s] \leftarrow \text{CV}[s] + \text{ChildCV}[j] \)
14: end for
15: end if

The example shown in Figure 4 illustrates the overall process. All the nodes within the AoI of sender \( n_{x} \) will be partitioned into three sets: \( \text{Core}_{v}, \text{Leaf}_{v} \) and \( U_{v}. \) When line 7 in Algorithm 1 is executed, \( \text{Core}_{v} = \{n_{a}, n_{y}, n_{z}\}. \) \( \text{Leaf}_{v} = \{n_{a}, n_{y}, n_{f}, n_{y}\}. \) \( U_{v} = \{n_{a}, n_{l}\}. \) Each node in \( \text{Core}_{v} \) and \( \text{Leaf}_{v} \) will be assigned a CV by Algorithm 3. Node \( n_{x} \) only has 1 available outdegree but which has two receivers, nodes \( n_{p} \) and \( n_{c}. \) It will choose the node which has the largest CV, node \( n_{y} \) in the example, as its receiver, and node \( n_{p} \) will be moved to \( U_{v}. \)
all-pairs-ping data from Planetlab as our training data set. The observed data and the well-fitting results are shown in Figure 7. Realistic in-coming and out-going bandwidth limits were obtained via random selection for each node from data gathered by dslreports.com (see Figure 8). The bandwidth for every audio stream was set to 25 kbps. The same Aoi radius was assigned to all nodes in the virtual world and we explored a range from 20 to 100 units. Figure 9 illustrates how the number of neighboring nodes is affected by their distribution in the virtual space as a function of the Aoi radius. More than 50 neighbors may be part of the Aoi in strongly clustered situations (Figures 9a and 10a) whereas it is usually less than 20 with a uniform distribution (Figures 9c and 10c).

To understand how well our peer-to-peer approach distributes audio to all the candidate receivers we first explored how the degree limits affected the number of receivers that could not be reached. We assume that 50% of the avatars in the virtual space are speaking at every time instance \( t \). Hence, each node across the whole virtual space has a random, 50% chance of being a speaker. Local multicast trees were constructed based on adapted minimum spanning tree (MST), minimum latency tree (MLT) and Hamiltonian path (HP) algorithms. Figure 10 illustrates their different in- and outdegree conflicts. Such a conflict occurs when a node exceeds its in- or outdegree limit. As expected the number of conflicts rises for highly clustered environments and with increasing Aoi radius. All algorithms experience the same indegree conflict rate since every node within the Aoi of a sender is supposed to receive the stream. HP has the least outdegree conflicts because each node has either zero or one node to forward its stream to. However, its average latency is high (see Table 4). MLT has the best latency due to a shallow tree height, but it results in the largest number of conflicts. Overall, a relatively low conflict rate can be achieved if the number of receivers is on the order of 20 and we have a high speaker density. With fewer speakers, larger receiver groups can be reached.

In our next experiment we computed the successful receiving ratio (SSR) defined as the number of nodes that successfully received their streams as a percentage of the nodes that should receive streams. Figure 11 shows that a high SSR can be achieved when a sender has not too many neighbors (note that the y-scales start at 50%). The graphs also illustrate that our stream mixing process can significantly boost the receiving ratio. Audio mixing may result in some spatial information loss because the position information of two or more audio sources are merged and averaged. Figure 12 demonstrates that the angular error introduced through this merging process is quite small and tolerable in most cases.

<table>
<thead>
<tr>
<th>Aoi radius</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLT</td>
<td>74.56</td>
<td>57.51</td>
<td>43.19</td>
<td>35.10</td>
<td>28.68</td>
</tr>
<tr>
<td>MST</td>
<td>77.34</td>
<td>64.67</td>
<td>53.97</td>
<td>47.86</td>
<td>42.08</td>
</tr>
<tr>
<td>HP</td>
<td>125.8</td>
<td>233.7</td>
<td>422.1</td>
<td>622.1</td>
<td>998.76</td>
</tr>
</tbody>
</table>

Table 4: Average latency using the MLT, MST, and HP local multicast tree algorithms (in ms).

5. CONCLUSIONS

We have introduced the design of a peer-to-peer architecture for the distribution of spatialized audio in networked virtual environments. The main motivation for our work is
the desire to create an audio experience that is more congruent with the three-dimensional visual display of virtual worlds while at the same time avoiding large, centralized server resources. Our initial results indicate that indeed, such a peer-to-peer topology is not only feasible but can produce good results. Our current work is mostly based on simulations and our plan going forward is to investigate the design in more depth via analytical analysis and a prototype implementation. For example, network traffic may be bursty. This could be due to variable voice traffic (e.g., the use of variable bit-rate compression and silence detection). On the other hand, the game traffic of an NVE can be bursty because the complexity of a scene changes, the user moves in the virtual world, etc. Such issues are quite challenging and can only be adequately addressed through measurements with a real implementation.

Several additional interesting aspects should be explored as well. For example, Boulstead et al. [3] have introduced a foreground versus background distinction that places more importance on the accurate sound rendering from close audio sources. This concept can also be incorporated into our design to further reduce the conflict rate. Additionally, more of the temporal dynamics of the system needs to be investigated. Among the relevant issues are peer churn (i.e., peers...
joining and leaving) and peer mobility (i.e., peers entering and leaving AoIs). Much of the required meta-information (such as avatar positions) is naturally distributed with the game control traffic. However, the challenges for the proposed algorithms are to adapt quickly and with reasonable computational complexity while keeping audio discontinuities (e.g., dropped samples or other distortions) to a minimum. We are currently working on extending our simulation model to study such peer dynamics and plan to perform realistic measurements with a prototype implementation.

6. ACKNOWLEDGMENTS

This research has been funded in part by NSF grant IIS-0534761, NUS AcRF grant WBS R-252-050-280-101/133 and A*STAR grant 082 101 0028. We also acknowledge the support of the NUS Interactive and Digital Media Institute (IDMI).

7. REFERENCES